

# 数据挖掘复习

## 概论

### ✓ 机器学习

机器学习在近30多年已发展为一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、计算复杂性理论等多门学科。机器学习理论主要是设计和分析一些让计算机可以自动“学习”的算法。机器学习算法是一类从数据中自动分析获得规律，并利用规律对未知数据进行预测的算法。因为学习算法中涉及了大量的统计学理论，机器学习与推断统计学联系尤为密切，也被称为统计学习理论。算法设计方面，机器学习理论关注可以实现的，行之有效的学习算法。

### ✓ 数据挖掘

数据挖掘（英语：data mining）是一个跨学科的计算机科学分支。它是用人工智能、机器学习、统计学和数据库的交叉方法在相对较大型的数据集中发现模式的计算过程。**数据挖掘过程的总体目标是从一个数据集中提取信息，并将其转换成可理解的结构，以进一步使用。**

数据挖掘是“数据库知识发现”的分析步骤。

### ✓ 机器学习和数据挖掘的关系

机器学习是数据挖掘的主要工具。

**数据挖掘**不仅仅要研究、拓展、应用一些机器学习方法，还要通过许多非机器学习技术解决数据存储、大规模数据、数据噪音等更为实际的问题。

**机器学习**的涉及面更宽，常用在数据挖掘上的方法通常只是“从数据学习”，然则机器学习不仅仅可以用在数据挖掘上，一些机器学习的子领域甚至与数据挖掘关系不大，例如**增强学习与自动控制**等。

大体上看，数据挖掘可以视为机器学习和数据库的交叉。

## ✓ 基本术语

### 泛化能力

机器学习的目标是使得学到的模型能很好的适用于“新样本”，而不仅仅是训练集合，我们称模型适用于新样本的能力为**泛化(generalization)能力**。

通常假设样本空间中的样本服从一个未知分布，样本从这个分布中独立获得，即“独立同分布”(i.i.d)。一般而言训练样本越多越有可能通过学习获得强泛化能力的模型。

### 监督学习

即样本是有标签的。

- 分类问题
- 回归问题
- 标注问题

监督学习目的是学习一个由输入到输出的映射，称为模型。模式的集合就是假设空间(hypothesis space)

### 半监督学习

- 少量标注数据，大量未标注数据
- 利用未标注数据的信息，辅助标注数据，进行监督学习
- 较低成本

### 主动学习

- 机器主动给出实例，教师进行标注
- 利用标注数据学习预测模型

## KNN

### 工作原理

- 存在一个样本数据集合，也称作**训练样本集**，样本集中每个数据都存在**标签**，即我们知道样本集中每个数据和所属分类
- 输入没有标签的新数据后，将**新数据的每个特征**与样本集中**数据**对应的**特征**进行比较，然后算法提取**样本集中特征最相似数据**（最近邻）的分类标签

- 一般来说，只选择样本数据集中前 k 个最相似的数据。最后，选择 k 个中出现次数最多的分类，作为新数据的分类

## 距离度量

**Distance functions**

Euclidean	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^k  x_i - y_i $
Minkowski	$\left( \sum_{i=1}^k ( x_i - y_i )^q \right)^{1/q}$

$K < \sqrt{n}$ , n is number of examples.

## 特点的归一化

$a_i = \frac{v_i - \min v_i}{\max v_i - \min v_i}$ , 使数据映射到 0-1。

## 如果考虑计算有序的和无序的属性的距离

1 hot 编码给无序的属性，有序的属性直接使用有序的编号。

## 特点

优点：

- 非常简单和直觉
- 能够适用于任何分布的数据
- 如果训练样本非常多，那么效果会非常好

缺点：

- 每分一个类，需要花大量时间
- k 的选择非常具有技巧性
- 需要大量的样本来提高精度

# 决策树

决策树是一种典型的分类方法。

- 决策树的优点
  - 1、推理过程易理解，决策推理过程可以表示成If Then形式；
  - 2、推理过程完全依赖于属性变量的取值特点；
  - 3、可自动忽略目标变量没有贡献的属性变量，也为判断属性变量的重要性，减少变量的数目提供参考。

## ✓ 算法

CLS, ID3, C4.5, CART

## ✓ CLS (concept learning system) 算法

### ID3 算法

是一种经典的决策树学习算法, 主要解决属性选择问题。

使用信息增益选择测试属性。

使用信息熵来度量样本集合纯度的常用指标，越大说明越不纯。

### 基本思想

ID3 算法的基本思想是，以**信息熵**为度量，用于决策树节点的属性选择，每次优先选取**信息量最多的属性**，亦即使熵值变为最小的属性，以构造**一颗熵值下降最快**的决策树，到叶子节点处的熵值为0。此时，每个叶子节点对应的实例集中的实例属于同一类。

## 划分选择-信息增益

- “信息熵”是度量样本集合纯度最常用的一种指标，假定当前样本集合  $D$  中第  $k$  类样本所占的比例为  $p_k$  ( $K = 1, 2, \dots, |\mathcal{Y}|$ )，则  $D$  的信息熵定义为

$$\text{Ent}(D) = - \sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k$$

$\text{Ent}(D)$  的值越小，则  $D$  的纯度越高

- 计算信息熵时约定：若  $p = 0$ ，则  $p \log_2 p = 0$
- $\text{Ent}(D)$  的最小值为 0，最大值为  $\log_2 |\mathcal{Y}|$

## 划分选择-信息增益

- 离散属性  $a$  有  $V$  个可能的取值  $\{a^1, a^2, \dots, a^V\}$ ，用  $a$  来进行划分，则会产生  $V$  个分支结点，其中第  $v$  个分支结点包含了  $D$  中所有在属性  $a$  上取值为  $a^v$  的样本，记为  $D^v$ 。则可计算出用属性  $a$  对样本集  $D$  进行划分所获得的“信息增益”：

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

为分支结点权重，样本数越多的分支结点的影响越大

- 一般而言，信息增益越大，则意味着使用属性  $a$  来进行划分所获得的“纯度提升”越大

存在问题：信息增益对可取值数目较多的属性有所偏好。

改进使用增益率：

□ 增益率定义:  $\text{Gain\_ratio}(D, a) = \frac{\text{Gain}(D, a)}{\text{IV}(a)}$

其中

$$\text{IV}(a) = - \sum_{v=1}^V \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

称为属性  $a$  的“固有值” [Quinlan, 1993], 属性  $a$  的可能取值数目越多 (即  $V$  越大), 则  $\text{IV}(a)$  的值通常就越大

□ 存在的问题

**增益率准则对可取值数目较少的属性有所偏好**

$\text{IV}(a)$  计算的是  $a$  这个属性各个可能的取值的一个信息熵? 比如  $a$  属性取 1, 2, 3 个数都是 1, 那么  $\text{IV}(a) = -3 \cdot (1/3) \log_2(1/3)$ 。

## C4.5 使用了一个启发式:

先从**候选划分属性**中找出**信息增益**高于**平均水平的属性**, 再从中选取**增益率最高**的。

## CART 树

这部分不是很清楚:

□ 数据集  $D$  的纯度可用 “基尼值” 来度量

$$\text{Gini}(D) = \sum_{k=1}^{|Y|} \sum_{k' \neq k} p_k p_{k'} = 1 - \sum_{k=1}^{|Y|} p_k^2$$

$\text{Gini}(D)$  越小, 数据集  $D$  的纯度越高

反映了从  $D$  中随机抽取两个样本, 其类别标记不一致的概率

□ 属性  $a$  的基尼指数定义为:

$$\text{Gini\_index}(D, a) = \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Gini}(D^v)$$

□ 应选择那个使划分后基尼指数最小的属性作为最优划分属性, 即

$$a_* = \underset{a \in A}{\text{argmin}} \text{Gini\_index}(D, a)$$

## ✓ 剪枝

理想的决策树有三种：

- 叶节点数目最少；
- 叶子节点的深度最小；
- 叶子节点树最少且叶子节点深度最小。

## 过拟合

剪枝可以减缓过拟合的问题。

- 预剪枝：

对每个结点在划分前先进行估计，若当前结点的划分不能带来决策树泛化性能提升，则停止划分并将当前结点记为叶结点，其类别标记为训练样例数最多的类别；

- 后剪枝：

先从训练集生成一棵完整的决策树，然后自底向上地对非叶结点进行考察，若将该结点对应的子树替换为叶结点能带来决策树泛化性能提升，则将该子树替换为叶结点。

## 数据缺失值处理

- 丢弃
- 使用缺失值的样本，主要要解决两个问题：
  - 如何在属性缺失的情况下进行划分属性选择？
    - 利用无缺失的样本来估计  $\text{Gain}(D,a)$ ，原理即为前面乘以无缺失的样本的比例。
  - 给定划分属性，若样本在该属性上的值缺失，如何对样本进行划分？
    - 如果已知，则直接划入。
    - 如果该属性上缺失，那么把该样本  $x$  同时划入所有分类的子结点中，并且修改该样本的权重（即该样本属于该类的概率，概率为不含缺失值划分下来到该类的频率/不含缺失值的总数）

# Ensembles 集成

集成学习(ensemble learning) 通过构建并结合多个学习器来提升性能;

集成多个模型的能力，达到比单一模型更佳的效果。这些算法可以是不同的算法，也可以是相同的算法。

## 个体与集成

- 关键性假设：基学习器的误差应该相互独立。
- 然而，现实任务中，个人学习器为解决同一个问题训练出来的，显然不可能相互独立。
- 个体学习器的 “准确性” 和 “多样性” 本身就存在冲突。
- 如何产生 “好而不同” 的个体学习器是集成学习研究的核心。
- 集成学习可以分为三类：Boosting, Bagging, Stacking

## Bagging

对样本或特征进行随机取样，学习产生多个独立的模型，然后平均所有模型的预测值。典型代表是随机森林。

### 随机森林

- 分类间隔(Margins): 随机森林的分类间隔是指森林中正确分类样本的决策树的比例减去错误样本决策树的比例。即正确 - 错误。

## Boosting

串行训练多个模型，后面的模型是基于前面模型的训练结果（误差）。它的代表是 AdaBoost;

### AdaBoost 算法

- AdaBoost 是 Boosting 算法的典型代表。核心思想是利用同一训练样本的不同加权版本，训练一组弱分类器，然后把这些弱分类器以加权的形式集成起来，形成一个最终的强分类器。
- 每一步迭代过程中，会给训练集中的样本赋予一个权重( $w_1, w_2, \dots, w_n$ )。样本的初始权重都一样，设置为  $\frac{1}{n}$ 。在每一步迭代过程中，被当前弱分类器分错的样本的权重会相应的得到提高，被当前弱分类器分对的样本的权重则会相应降低。弱分类器的权重则根据当前分类器的**加权错误率**来确定。



# Stacking

多层学习的味道，有点像深度学习。

- **Stacking**的具体过程如下：
  1. 划分训练数据集为两（或多）个不相交的集合。
  2. 在第一个集合上训练多个学习器。
  3. 在第二个集合上测试这几个学习器
  4. 把第三步得到的预测结果作为输入，把正确的回应作为输出，训练一个高层学习器

## 机器学习的组合策略的好处

### 结合策略

平均法，连续标签时

- 简单平均法
- 加权平均法

投票法，离散标签时

- 绝对多数投票法
- 相对多数投票法
- 加权投票法

学习法

- Stacking 是学习法的典型代表

### 多样性-多样性增强

- 常见的增强个体学习器的多样性的方法
  - 数据样本扰动：
    - 基于采样法
  - 输入属性扰动

# 线性模型

## ✓ 回归

是一种模型，简单来说就是需要预测的变量是连续值。

## ✓ 基本形式

### 基本形式

- 线性模型一般形式

$$f(\mathbf{x}) = w_1x_1 + w_2x_2 + \dots + w_dx_d + b$$

$\mathbf{x} = (x_1; x_2; \dots; x_d)$  是由属性描述的示例，其中  $x_i$  是  $\mathbf{x}$  在第  $i$  个属性上的取值

- 向量形式  $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$

其中  $\mathbf{w} = (w_1; w_2; \dots; w_d)$

## ✓ 线性回归

# 线性回归

- 给定数据集  $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_m, y_m)\}$   
其中  $\mathbf{x}_i = (x_{i1}; x_{i2}; \dots; x_{id})$  ,  $y_i \in \mathbb{R}$
- 线性回归 (linear regression) 目的
  - 学得一个线性模型以尽可能准确地预测实值输出标记
- 离散属性处理
  - 有“序”关系
    - 连续化为连续值
  - 无“序”关系
    - 有k个属性值, 则转换为k维向量

- 单一属性的线性回归目标

$$f(x) = wx_i + b \text{ 使得 } f(x_i) \simeq y_i$$

- 参数/模型估计: 最小二乘法 (least square method)

$$\begin{aligned}(w^*, b^*) &= \arg \min_{(w, b)} \sum_{i=1}^m (f(x_i) - y_i)^2 \\ &= \arg \min_{(w, b)} \sum_{i=1}^m (y_i - wx_i - b)^2\end{aligned}$$

最小二乘法即 OK。求导即可。

## ✓ 逻辑回归

二分类任务。

标签  $y \in \{0, 1\}$ 。

使用的函数为 sigmoid

$$y = \frac{1}{1 + e^{-z}}$$

运用逻辑函数变成逻辑回归：

$$y = \frac{1}{1 + e^{-(w^T x + b)}}$$

使用对数几率 (log odds) 来表示样本作为**正例**的相对可能性的对数。

$$\ln \frac{y}{1 - y}$$

逻辑回归优点

- 无需事先假设数据分布
- 可得到“类别”的近似概率预测
- 可直接应用现有数值优化算法求取最优解

## 极大似然法

- 对数几率

$$\ln \frac{p(y = 1 | \mathbf{x})}{p(y = 0 | \mathbf{x})} = \mathbf{w}^T \mathbf{x} + b$$

**逻辑回归是一种广义线性模型**

显然有

$$p(y = 1 | \mathbf{x}) = \frac{e^{\mathbf{w}^T \mathbf{x} + b}}{1 + e^{\mathbf{w}^T \mathbf{x} + b}}$$

$$p(y = 0 | \mathbf{x}) = \frac{1}{1 + e^{\mathbf{w}^T \mathbf{x} + b}}$$

即所有样本预测对的**概率乘积**最大。

## 极大似然法 (maximum likelihood)

- 给定数据集  $\{(\mathbf{x}_i, y_i)\}_{i=1}^m$
- 最大化样本属于其真实标记的概率
  - 最大化对数似然函数

$$\ell(\mathbf{w}, b) = \sum_{i=1}^m \ln p(y_i | \mathbf{x}_i; \mathbf{w}_i, b)$$

## ✓ 二分类任务——线性判别分析

线性判别分析 (Linear Discriminant Analysis)

- LDA的思想：给定训练样例集，设法将样例投影到一条直线上，使得同类样例的投影点尽可能接近，不同类的投影点尽可能远离；
- 在对新样本进行分类时，将其投影到这条直线上，根据投影点的位置确定新样本的类别；
- 一种有监督降维技术

## ✓ 多分类学习

- 多分类学习方法
  - 二分类学习方法推广到多类
  - 利用二分类学习器解决多分类问题
    - 对问题进行拆分，为拆出的每个二分类任务训练一个分类器

## SVM 支持向量机

线性模型：在样本空间中寻找一个超平面，将不同类别的样本分开。

相关表示：

- 超平面方程：

$$\mathbf{w}^T \mathbf{x} + b = 0$$

假设这个平面在两个类的支持向量的中间，然后要使得分类间隔最大，则间隔计算公式如下：

$$\gamma = \frac{2}{\|\mathbf{w}\|}$$

- 寻找最大间隔，所以目标函数为：

$$\begin{aligned} & \arg_{\mathbf{w}, b} \max \frac{2}{\|\mathbf{w}\|} \\ & s.t. y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 \end{aligned}$$

- 对偶问题

## 对偶问题

KKT条件：

$$\begin{cases} \alpha_i \geq 0, \\ y_i f(\mathbf{x}_i) \geq 1, \\ \alpha_i (y_i f(\mathbf{x}_i) - 1) = 0. \end{cases}$$

### □ 拉格朗日乘子法

- 第一步：引入拉格朗日乘子  $\alpha_i \geq 0$  得到拉格朗日函数

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^m \alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1)$$

- 第二步：令  $L(\mathbf{w}, b, \boldsymbol{\alpha})$  对  $\mathbf{w}$  和  $b$  的偏导为零可得

$$\mathbf{w} = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i, \quad \sum_{i=1}^m \alpha_i y_i = 0.$$

- 第三步：回代

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \sum_{i=1}^m \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \quad \alpha_i \geq 0, \quad i = 1, 2, \dots, m. \end{aligned}$$

149

- 最终模型：

- 最终模型：  $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^m \alpha_i y_i \mathbf{x}_i^T \mathbf{x} + b$

支持向量机解的稀疏性：训练完成后，大部分的训练样本都不需保留，最终模型仅与支持向量有关。

- 求解方法：SMO，注意求解的问题为对偶问题的第三步的问题形式，求解  $\alpha$

# SVM 的核函数，SVM 的翅膀

若不存在一个能正确划分两类样本的超平面, 怎么办?

- 将样本从原始空间映射到一个更高维的特征空间, 使得样本在这个特征空间内线性可分.

## 核支持向量机

□ 设样本  $\mathbf{x}$  映射后的向量为  $\phi(\mathbf{x})$ , 划分超平面为  $f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b$

原始问题

$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{1}{2} \|\mathbf{w}\|^2 \\ \text{s.t.} \quad & y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) \geq 1, \quad i = 1, 2, \dots, m. \end{aligned}$$

对偶问题

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) - \sum_{i=1}^m \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \quad \text{只以内积的形式出现} \end{aligned}$$

预测

$$f(\mathbf{x}) = \mathbf{w}^\top \phi(\mathbf{x}) + b = \sum_{i=1}^m \alpha_i y_i \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}) + b$$

- 基本思想:
  - 不显式地设计核映射, 而是设计核函数

## 核函数

- 基本想法: 不显式地设计核映射, 而是设计核函数.

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

- Mercer定理(充分非必要): 只要一个对称函数所对应的核矩阵半正定, 则它就能作为核函数来使用.
- 常用核函数:

名称	表达式	参数
线性核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$	
多项式核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^\top \mathbf{x}_j)^d$	$d \geq 1$ 为多项式的次数
高斯核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ ^2}{2\delta^2}\right)$	$\delta > 0$ 为高斯核的带宽(width)
拉普拉斯核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\ \mathbf{x}_i - \mathbf{x}_j\ }{\delta}\right)$	$\delta > 0$
Sigmoid核	$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta \mathbf{x}_i^\top \mathbf{x}_j + \theta)$	$\tanh$ 为双曲正切函数, $\beta > 0, \theta < 0$

## ✓ 软间隔

现实中很难找到一个超平面使得样本完全线性可分，有可能是过拟合造成的，所以引入软间隔的概念：允许支持向量机上有一些样本不满足约束条件。

修改目标函数为：

- 基本想法：最大化间隔的同时，让不满足约束的样本应尽可能少。

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m l_{0/1}(y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b) - 1)$$

其中  $l_{0/1}$  是“0/1损失函数”

$$l_{0/1} = \begin{cases} 1 & z < 0 \\ 0 & otherwise \end{cases}$$

- 存在的问题：0/1损失函数非凸、非连续，不易优化！

## 软间隔支持向量机

原始问题  $\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m \max(0, 1 - y_i(\mathbf{w}^\top \phi(\mathbf{x}_i) + b))$

对偶问题 
$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) - \sum_{i=1}^m \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^m \alpha_i y_i = 0, \quad 0 \leq \alpha_i \leq C, \quad i = 1, 2, \dots, m. \end{aligned}$$

根据KKT条件可推得最终模型仅与支持向量有关，也即hinge损失函数依然保持了支持向量机解的稀疏性。



# 深度学习

## 感知机

一层感知机：即只有一层隐藏层，即只激活一次。

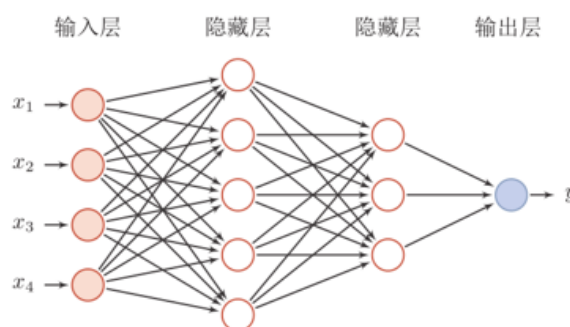
- 同一层的神经元不互连
- 不同层之间的神经元全连接

## 近似定理

在人工神经网络的数学理论中，通用近似定理，也称万能近似定理（universal approximation theorem）指出，对于具有线性输出层和至少一个使用“挤压”性质的激活函数（例如sigmoid激活函数）的隐藏层组成的前馈神经网络，它可以以任意的精度来近似任何从一个定义在实数空间中的有界闭集函数。

## ✓ 前馈神经网络

- 前馈神经网络（全连接神经网络、多层感知机）
  - 各神经元分别属于不同的层，层内无连接。
  - 相邻两层之间的神经元全部两两连接。
  - 整个网络中无反馈，信号从输入层向输出层单向传播，可用一个有向无环图表示。



## Transformer @有时间再看

- 循环神经网络的缺点

- 循环结构难以并行化处理
- 难以捕捉长期依赖
- Transformer 完全基于注意力机制

## 特征选择

### ✓ Introduction

特征选择

- 从给定的特征集合中选出**任务相关特征子集**
- 必须确保不丢失重要特征

原因：

- 减轻维度灾难：在少量属性上构建模型
- 降低学习难度：留下关键信息
- 提高模型的预测准确率，即可以防止过拟合
- 可以构造速度更快、消耗更低的预测模型
- 能够对模型有更好的解释性

特征选择：

- 从原始特征中挑选出一些最有代表性，性能最好的特征

特征提取：

- 用**映射**的方法把**原始特征**变换为**较少的新特征**

特征选择和数据降维的关系

- 都通过减少**特征数量**来**降低模型复杂度**，从而防止过度拟合
- 降维通过**原始特征**构造**新的特征**，新的特征能够更好地表示原始数据，而且特征数量较原始特征少
- 特征选择并不构造新的特征，它从原始特征中选取若干特征来表示原始数据

### ✓ 特征选择的一般过程

## 可行方法



两个关键环节：子集搜索和子集评价

## 子集搜索

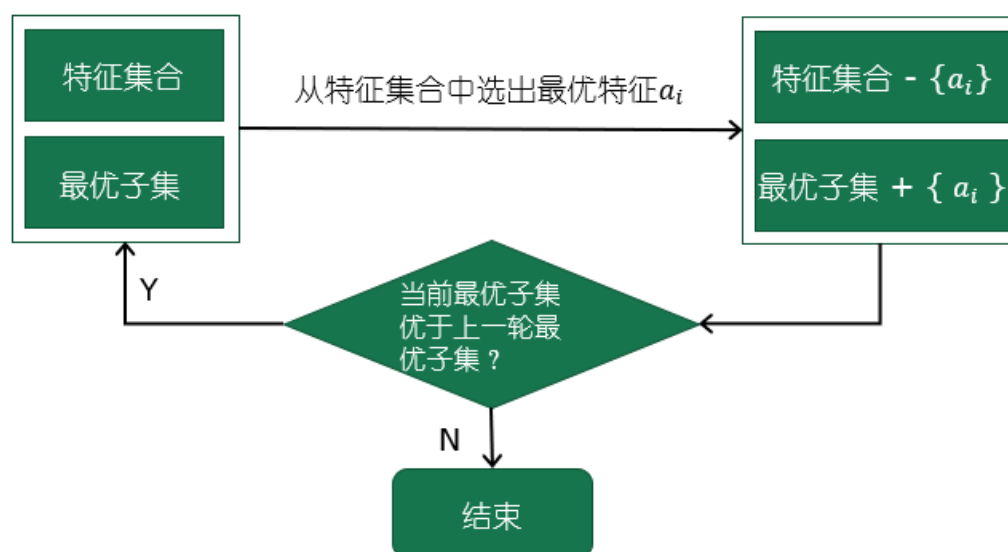
用贪心策略选择包含重要信息的特征子集

- 前向搜索：逐渐增加相关特征
- 后向搜索：从完整的特征集合开始，逐渐减少特征
- 双向搜索：每一轮逐渐增加相关特征，同时减少无关特征

## 前向搜索

最优子集初始为空集，特征集合初始时包括所有给定特征。

□ 最优子集初始为空集，特征集合初始时包括所有给定特征



## 子集评价

特征子集确定了对数据集的一个划分

- 每个划分区域对应着特征子集的某种取值

样本标记对应着对数据集的真实划分。

评价方法：通过估算两个划分的差异，与样本标记的划分的差异越小，则说明当前子集越好。

比如可以利用信息熵来衡量：

□ 特征子集 $A$ 确定了对数据集 $D$ 的一个划分

- $A$ 上的取值将数据集 $D$ 分为 $V$ 份，每一份用 $D^v$ 表示
- $\text{Ent}(D^v)$ 表示 $D^v$ 上的信息熵

□ 样本标记 $Y$ 对应着对数据集 $D$ 的真实划分

- $\text{Ent}(D)$ 表示 $D$ 上的信息熵

$D$ 上的信息熵定义为

$$\text{Ent}(D) = - \sum_{i=1}^Y p_i \log_2 p_i$$

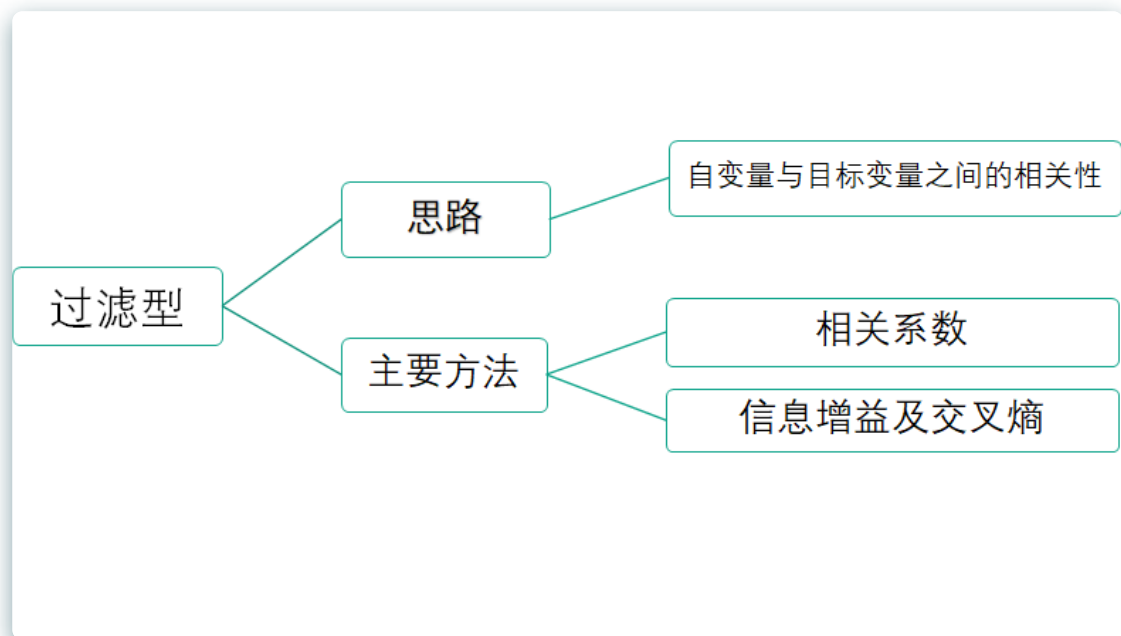
第 $i$ 类样本所占比例为 $p_i$

特征子集 $A$ 的信息增益为：

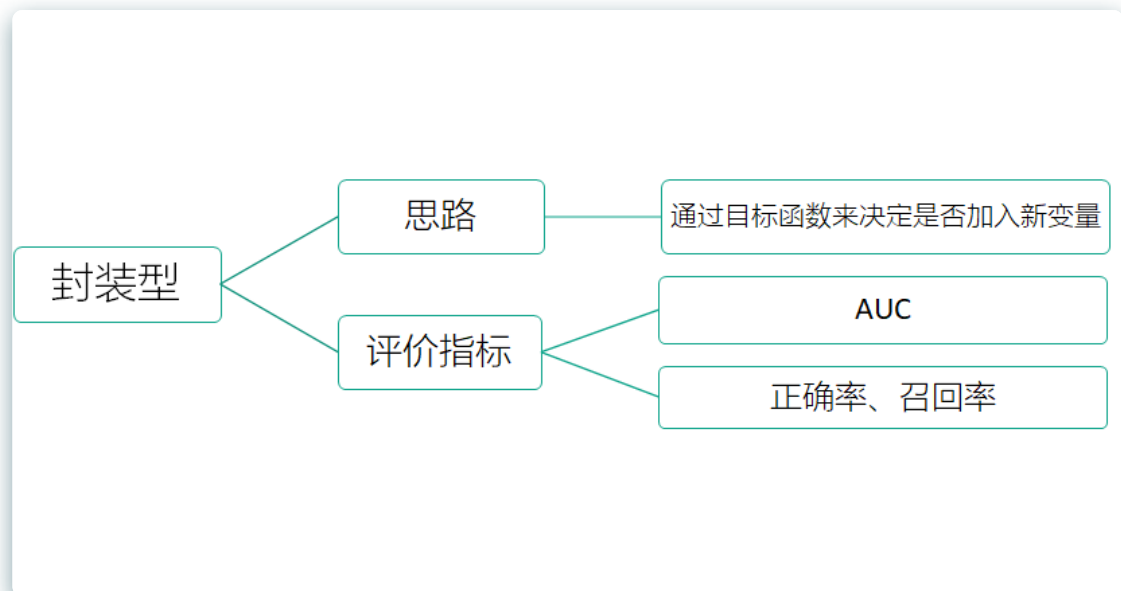
$$\text{Gain}(A) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v)$$

## ✓ 常见的特征选择方法

过滤型方法：



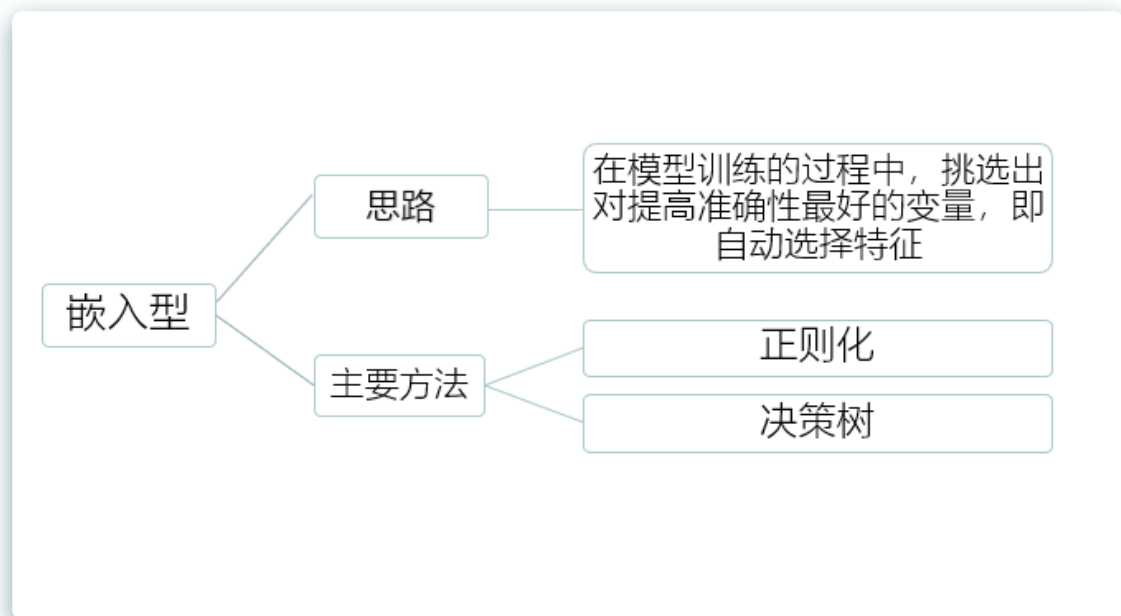
### 封装型方法：



封装型方法直接把**学习器的性能**作为特征子集的评价标准。

其实就是随机选一些属性，然后训练一个模型看看结果。

### 嵌入型方法：



这是一种把特征选择和模型训练融为一体的方法，二者在同一个优化的过程中完成。

- 正则化项
- 基于树的模型：
  - 基于树的模型可以输出特征的重要性
    - 根据该指标筛选出重要的特征从而达到特征选择的效果
  - 平均不纯度减少
    - 决策树模型可以计算出每个特征减少了多少的不纯度，比如信息增益
  - 平均精度率减少
    - 另一种常用方法是度量每个特征对模型准确率的影响：
      - 打乱每个特征的特征值的顺序，并且度量顺序变动对模型的准确率的影响
- 递归特征消除：反复构建模型

## ✓ 无监督特征选择

有监督的特征提取方法使用特征与样本标签之间的相关性来评估特征的重要性

- 在很多情况下，标注数据的代价是高昂的，很难在大数据集上进行，无监督的特征选择方法就显得尤为重要

由于没有标签信息

- **无监督特征选择**利用**样本间的相似性**和**局部判别信息**来定义特征间的相关性

# PCA

Principal Components Analysis 主成分分析

为什么数据能进行降维？

- 数据样本虽然是高维的，但与学习任务密切相关的也许仅是某个低维分布，即高维空间中的一个低维 “嵌入”，因而可以对数据进行有效的降维。

## ✓ 线性降维方法

乘以一个矩阵  $W$ ,

$$Z = W^T X, W \in \mathbb{R}^{d \times d'} \\ Z \in \mathbb{R}^{d' \times m}$$

其中  $W$  是变换矩阵。

如果假定  $W$  中各列元素正交，那么构成一个正交系。

PCA 的思想是：

- 找到这样一个正交的新空间  $W$ ，即超平面，使得所有样本映射到平面式满足：
  - 最近重构性：样本点到这个超平面的举例都足够近
  - 最大可分性：样本点到这个超平面上的投影尽可能分开

## ✓ 最大可分性

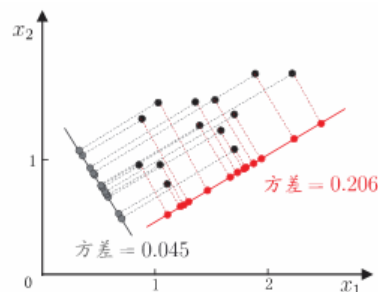
尽可能分开，等价于投影后的样本的方差最大化。

## 最大可分性

□ 样本点  $\mathbf{x}_i$  在新空间中超平面上的投影是  $\mathbf{W}^T \mathbf{x}_i$ ，若所有样本点的投影能尽可能分开，则应该使得投影后样本点的方差最大化。若投影后样本点的方差是  $\sum_i \mathbf{W}^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{W}$ ，于是优化目标可写为

$$\begin{aligned} \max_{\mathbf{W}} \quad & \text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) \\ \text{s.t.} \quad & \mathbf{W}^T \mathbf{W} = \mathbf{I}. \end{aligned}$$

显然与  $\min_{\mathbf{W}} -\text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W})$  等价。  
s.t.  $\mathbf{W}^T \mathbf{W} = \mathbf{I}.$



求解上述优化问题：

## PCA的求解

$$\begin{aligned} \max_{\mathbf{W}} \quad & \text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) \\ \text{s.t.} \quad & \mathbf{W}^T \mathbf{W} = \mathbf{I}. \end{aligned}$$

$$\begin{aligned} \min_{\mathbf{W}} \quad & -\text{tr}(\mathbf{W}^T \mathbf{X} \mathbf{X}^T \mathbf{W}) \\ \text{s.t.} \quad & \mathbf{W}^T \mathbf{W} = \mathbf{I}. \end{aligned}$$

□ 对优化式使用拉格朗日乘子法可得

$$\mathbf{X} \mathbf{X}^T \mathbf{W} = \lambda \mathbf{W}.$$

只需对协方差矩阵  $\mathbf{X} \mathbf{X}^T$  进行特征值分解，并将求得的特征值排序： $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d$ ，再取前  $d'$  个特征值对应的特征向量构成  $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'})$ ，这就是主成分分析的解。

其中降维后的空间维数  $d'$  通常是由用户事先指定。对 PCA, 还可以从重构的角度设置一个重构阈值，例如  $t = 95\%$ ，然后选取使下式成立的最小  $d'$  值：



$$\frac{\sum_{i=1}^{d'} \lambda_i}{\sum_{i=1}^d \lambda_i} \geq t.$$

## 算法如下

注: 样本需要先中心化

### PCA算法

---

输入: 样本集  $D = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ ;  
低维空间维数  $d'$ .

过程:

- 1: 对所有样本进行中心化:  $\mathbf{x}_i \leftarrow \mathbf{x}_i - \frac{1}{m} \sum_{i=1}^m \mathbf{x}_i$ ;
- 2: 计算样本的协方差矩阵  $\mathbf{XX}^T$ ;
- 3: 对协方差矩阵  $\mathbf{XX}^T$  做特征值分解;
- 4: 取最大的  $d'$  个特征值所对应的特征向量  $\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'}$ .

输出: 投影矩阵  $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_{d'})$ .

---

## ✓ 核化线性降维 Kernelized PCA

线性降维方法假设高维空间到低维空间的函数映射是线性的, 然而, 实际中可能需要非线性映射才能找到恰当的低维嵌入。

## 核化主成分分析 (Kernelized PCA, 简称KPCA)

□ 非线性降维的一种常用方法，是基于核技巧对线性降维方法进行“核化” (kernelized)。

□ 假定我们将在高维特征空间中把数据投影到由  $\mathbf{W}$  确定的超平面上，即PCA欲求解

$$\left( \sum_{i=1}^m \mathbf{z}_i \mathbf{z}_i^T \right) \mathbf{W} = \lambda \mathbf{W}.$$

□ 其中  $\mathbf{z}_i$  是样本点  $\mathbf{x}_i$  在高维特征空间中的像。令  $\boldsymbol{\alpha}_i = \frac{1}{\lambda} \mathbf{z}_i^T \mathbf{W}$ ,

$$\mathbf{W} = \frac{1}{\lambda} \left( \sum_{i=1}^m \mathbf{z}_i \mathbf{z}_i^T \right) \mathbf{W} = \sum_{i=1}^m \mathbf{z}_i \frac{\mathbf{z}_i^T \mathbf{W}}{\lambda} = \sum_{i=1}^m \mathbf{z}_i \boldsymbol{\alpha}_i.$$

## 核化主成分分析 (Kernelized PCA, 简称KPCA)

□ 假定  $\mathbf{z}_i$  是由原始属性空间中的样本点  $\mathbf{x}_i$  通过映射  $\phi$  产生，即

$$\mathbf{z}_i = \phi(\mathbf{x}_i), \quad i = 1, 2, \dots, m.$$

□ 若  $\phi$  能被显式表达出来，则通过它将样本映射至高维空间，再在特征空间中实施PCA即可，即有

$$\left( \sum_{i=1}^m \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T \right) \mathbf{W} = \lambda \mathbf{W}.$$

并且

$$\mathbf{W} = \sum_{i=1}^m \phi(\mathbf{x}_i) \boldsymbol{\alpha}_i.$$

□ 一般情形下，我们不清楚  $\phi$  的具体形式，于是引入核函数

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j).$$

□ 又由  $\mathbf{W} = \sum_{i=1}^m \phi(\mathbf{x}_i) \boldsymbol{\alpha}_i$ ，代入优化式  $\left( \sum_{i=1}^m \phi(\mathbf{x}_i) \phi(\mathbf{x}_i)^T \right) \mathbf{W} = \lambda \mathbf{W}$ ，有

$$\mathbf{K} \mathbf{A} = \lambda \mathbf{A}.$$

其中  $\mathbf{K}$  为  $\kappa$  对应的核矩阵,  $(\mathbf{K})_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$ ,  $\mathbf{A} = (\boldsymbol{\alpha}_1; \boldsymbol{\alpha}_2; \dots; \boldsymbol{\alpha}_m)$ .

□ 上式为特征值分解问题，取  $\mathbf{K}$  最大的  $d'$  个特征值对应的特征向量得到解。

## 核化主成分分析 (Kernelized PCA, 简称KPCA)

□ 对新样本  $\mathbf{x}$ , 其投影后的第  $j$  ( $j = 1, 2, \dots, d'$ ) 维坐标为

$$\begin{aligned} z_j &= \mathbf{w}_j^T \phi(\mathbf{x}) = \sum_{i=1}^m \alpha_i^j \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) \\ &= \sum_{i=1}^m \alpha_i^j \kappa(\mathbf{x}_i, \mathbf{x}). \end{aligned}$$

其中  $\alpha_i$  已经过规范化,  $\alpha_i^j$  是  $\alpha_i$  的第  $j$  个分量。由该式可知, 为获得投影后的坐标, KPCA需对所有样本求和, 因此它的计算开销较大。

## ✓ 流形学习

- 借鉴拓扑流形概念的降维方法。“流形”在局部具有欧式空间的性质, 能用欧式距离来进行距离计算。
- 若低维流形嵌入到高维空间中, 可以在局部上保留欧式空间的性质。
- 流形学习可以用于可视化。

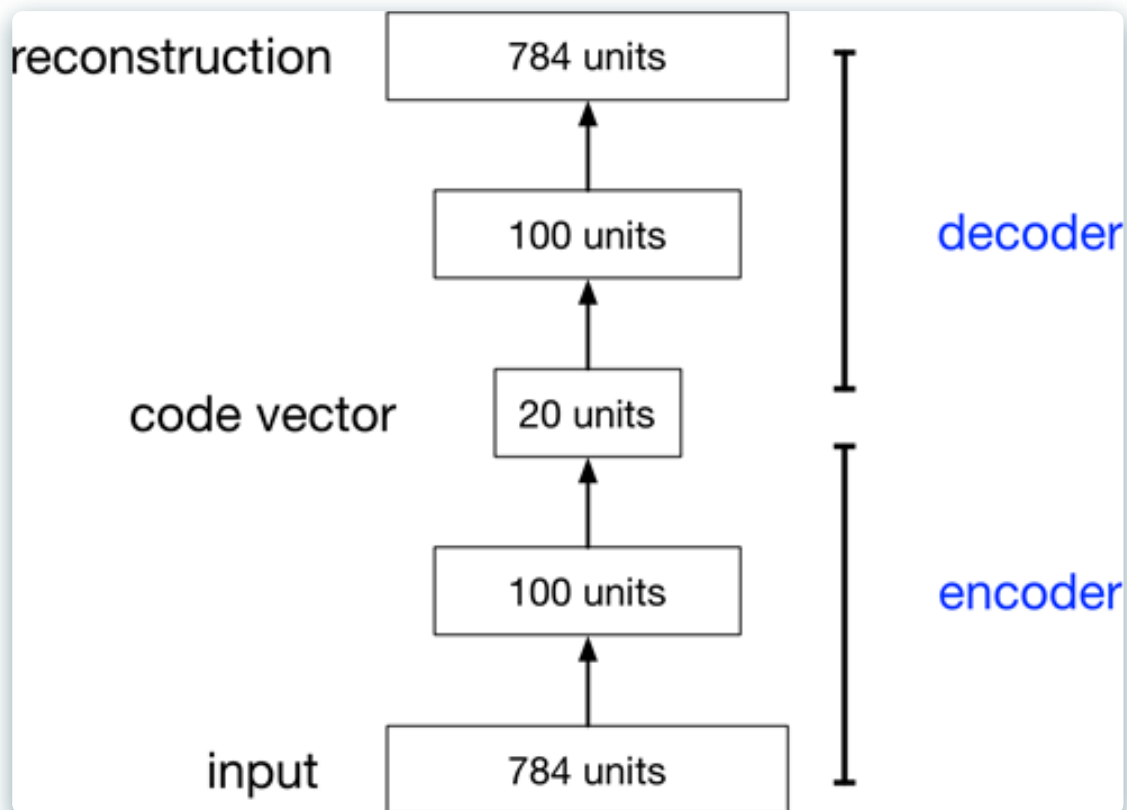
## 局部线性嵌入 (LLE)

总之就是在高维空间内找到某个点与它周围领域内的点的权重关系  $w$ 。然后在低维空间下, 这些权重信息可以使用。

## ✓ 自编码

通过神经网络的方法, 输入为原数据样本, 输出的标签也为原数据样本。

网络如下: encoder 和 decoder



## 聚类——无监督学习

无监督学习任务中研究最多，应用最广。

### □ 形式化描述

假定样本集  $D = \{x_1, x_2, \dots, x_m\}$  包含  $m$  个无标记样本, 每个样本  $x_i = (x_{i1}; x_{i2}; \dots; x_{in})$  是一个  $n$  维的特征向量, 聚类算法将样本集  $D$  划分成  $k$  个不相交的簇  $\{C_l | l = 1, 2, \dots, k\}$ 。其中  $C_{l'} \cap_{l' \neq l} C_l = \phi$ , 且  $D = \bigcup_{l=1}^k C_l$ 。

相应地, 用  $\lambda_j \in \{1, 2, \dots, k\}$  表示样本  $x_j$  的“簇标记” (即cluster label), 即  $x_j \in C_{\lambda_j}$ 。于是, 聚类的结果可用包含  $m$  个元素的簇标记向量  $\lambda = \{\lambda_1; \lambda_2; \dots; \lambda_m\}$  表示。

- 聚类性能度量, 亦称为聚类“有效性指标” (validity index)
  - 外部指标 (external index)
    - 将聚类结果与某个“参考模型”进行比较

- 内部指标 (internal index)

- 直接考察聚类结果而不用任何参考模型

- 一些指标:

- 考虑聚类结果的簇划分  $C = \{C_1, C_2, \dots, C_k\}$ , 定义:

- 簇  $C_l$  内样本间的平均距离

$$avg(C_l) = \frac{2}{|C_l|(|C_l|-1)} \sum_{1 \leq i \leq j \leq |C_l|} dist(x_i, x_j)$$

- 簇  $C_l$  内样本间的最远距离

$$diam(C_l) = \max_{1 \leq i \leq j \leq |C_l|} dist(x_i, x_j)$$

- 簇  $C_i$  与簇  $C_j$  最近样本间的距离

$$d_{min}(C) = \min_{x_i \in C_i, x_j \in C_j} dist(x_i, x_j)$$

- 簇  $C_i$  与簇  $C_j$  中心点间的距离

$$d_{cen}(C) = dist(\mu_i, \mu_j)$$

## ✓ 原型聚类 —— k均值算法

将样本分为 k 类, 目标是最小化平方误差:

$$E = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|_2^2$$

其中,  $\mu_i$  是簇  $C_i$  的均值向量。

## 算法

## □ 算法流程（迭代优化）：

初始化每个簇的均值向量

**repeat**

1. （更新）簇划分；
2. 计算每个簇的均值向量

**until** 当前均值向量均未更新

## 半监督学习

样本只有少量有标记的情况，大量标签缺失的情况。

### ✓ 生成式方法

利用高斯混合模型（EM）模型，直接假设数据的分布是一个高斯分布，然后利用 EM 模型来求解该高斯分布的参数，确定之后就可以预测样本的标签。

- 适用于数据极少的情况
- 模型假设必须准确

### ✓ 半监督 SVM

使用 SVM 方法来建立模型。先通过**有标签**的样本信息训练 SVM 模型，然后通过该训练得到的模型来**预测无标签样本的标签**，之后再**输入模型**再次训练。

### ✓ 图半监督学习

- 给定一个数据集，我们可将其映射为一个图，数据集中每个样本对应于图中一个结点，若两个样本之间的相似度很高(或相关性很强)，则对应的结点之间存在一条边，边的“强度” (strength)正比于样本之间的相似度(或相关性)。

- 把数据信息建模成图，样本为图的节点，**节点之间的边**表示**样本之间的相似性**。然后**带标签的节点**通过边把**标签信息扩散**给**不含标记的节点**。

## ✓ 半监督聚类

这是 k 均值算法的扩展,它在聚类过程中要**确保“必连”关系集合与“勿连”关系集合中的约束**得以 满足。与上面三个方法不同是，**聚类本身就是无监督学习**，引入有标签的样本实际上是通过**监督学习的形式**来**增强无监督学习的效果**，刚好与上面三个学习方法相反。

## 推荐系统

### ✓ 模型形式化

$X$  = set of Customers

$S$  = set of Items

Utility function  $u: X * S \rightarrow R$

- $R$  = set of ratings
- $R$  is a totally ordered set
- e.g. 0-5 stars, real number in  $[0,1]$

Utility Matrix 效用矩阵

### ✓ 关键问题

- 从 matrix 中获取已知的 rating，获取关联？
  - 怎样从 utility matrix 收集数据

- 从已知评级中推断未知评级

Mainly interested in high unknown ratings

- We are not interested in knowing what you don't like but what you like
- 评估推断方法
  - 怎样评估推荐的好坏？

## ✓ 获取评级

显式

- 叫用户打分
- 众包

隐式

- 从其他用户处学习

## ✓ 扩充? 推断 Utilities

冷启动的问题:

- 没有用户的历史消息
- 新的商品没有数据怎么办?

## 解决办法

### 基于内容的推荐系统

推荐相似的内容的东西给用户。

- 创建商品的 profile
  - 每个项创建一个 profile
  - 商品需要一些特征

如果寻找商品的特征?

- TF-IDF
- 创建用户 profile
  - 用户肖像概率, 可能性
- 预测的启发式函数为:

- 用户肖像为  $x$ , item 的肖像为  $i$ , 则评价 estimate 为:

$$u(x, i) = \cos(x, i) = \frac{x \cdot i}{||x|| \cdot ||i||}$$



## 优点

- 不需要其他用户的数据
- 可以推荐新的电影

## 缺点

- 找到合适的特征不太容易
- Overspecialization。

## 协同过滤

基于 user 到 user 的推荐。

考虑用户  $x$

- 找  $N$  个其他用户，他们之间的评分和  $x$  的 rating 很相近
- 根据  $N$  个用户来 estimate  $x$  的 rating

### 找相似的用户

设  $r_x$  表示用户  $x$  的 ratings

相似度采用余弦相似度:  $\text{sim}(x,y) = \arccos(r_x, r_y) = \frac{r_x \cdot r_y}{\|r_x\| \cdot \|r_y\|}$

### rating 预测

- $r_x$  是用户  $x$  的 ratings
- $N$  是最和  $x$  相似的  $k$  个用户对应的某一项的下标集合

所以预测  $x$  的评分为：

$$r_{xi} = (1/k) \sum_{y \in N} r_{yi}$$

## 基于 item - item 的协同过滤


- 对于 item  $i$ , 找它相似的 items
- 估计 item  $i$  的 rating 根据其他相似的项
- 可以使用这样的估计函数：

$$r_{xi} = \frac{\sum_{j \in N(i;x)} s_{ij} \cdot r_{xj}}{\sum_{j \in N(i;x)} s_{ij}}$$

$s_{ij}$ ... similarity of items  $i$  and  $j$   
 $r_{xj}$ ... rating of user  $u$  on item  $j$   
 $N(i;x)$ ... set items rated by  $x$  similar to  $i$

举个例子：

		users											
		1	2	3	4	5	6	7	8	9	10	11	12
movies	1	1		3		?	5			5		4	
	2			5	4			4			2	1	3
	3	2	4		1	2		3		4	3	5	
	4		2	4		5			4			2	
	5			4	3	4	2					2	5
	6	1		3		3			2			4	

 - estimate rating of movie 1 by user 5

现在估计 ? 处。

然后一行一行看，一行就是一个 item。

然后来计算电影 1 和其他电影的相似度。得到一系列相似度权重  $w$ 。

然后再来算 user 5 对电影1的评分：

- 根据user5 的其他的评分，然后利用权重  $w$  来加权，得到最终的评分。

## Common Practice 经验

$r_{xi}$  表示的是用户  $x$  对  $i$  item 的打分。

- 定义相似度  $s_{ij}$  为 item  $i$  和 item  $j$ 的相似度
- 选择  $k$  个最近的邻居  $N(i; x)$ 
  - 和 item  $i$  最近的 items, 被用户  $x$  打分
- 估计  $r_{xi}$  的式子如下：

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for  $r_{xi}$

$$b_{xi} = \mu + b_x + b_i$$

- $\mu$  = overall mean movie rating
- $b_x$  = rating deviation of user  $x$   
= (avg. rating of user  $x$ ) -  $\mu$
- $b_i$  = rating deviation of movie  $i$

$b_{xi}$  = 该item  $i$  的平均评分 + 用户  $x$  的打分偏差 + 该 item  $i$  的打分偏差。

## ✓ 评价方法

- 挖去矩阵的一部分
- 使用 RMSE, 和已知评分来做比较。

	movies					
users	1	3	4			
		3	5			5
			4	5		5
			3			
			3			
	2			?		?
					?	
		2	1			?
		3			?	
	1					

Test Data Set